Using the Cross-Entropy Method for Control Optimization: A Case Study of See-and-Avoid on Unmanned Aerial Vehicles

Miguel A. Olivares-Mendez Changhong Fu Somasundar Kannan Holger Voos Pascual Campoy

I. INTRODUCTION

The control issue is one of the most important aims in the industry, an also in robotics. In the specific case of the aerial vehicles, the complexity increases comparing them to ground and underwater robots. In the UAV's case, the control system have to deal with stability, wind disturbances and other forces just to keep the robot hovering in an specific location. The uncertainty, inaccuracy, approximation and incompleteness problems of the extracted data from the sensors have to be taken into account to close a control loop. These issues could be deal more easily with some of the Soft Computing (SC) approaches. Fuzzy Logic Controller (FLC) is one of the most active and fruitful SC control techniques. This technique use the expert knowledge to configure the controller, but a fine tuning is also needed to adjust the controller for the specific problem to solve. It could be done manually or using an autonomous optimization method. The optimization of this type of controllers for UAVs is already presented in the literature. Some of the most relevant of works are [1], that shows a self-tunable fuzzy inference system (STFIS) compared with model-based control system to control a drone in presence of disturbances. The classical and multi-objective genetic algorithm (GA) based fuzzy-genetic autopilot are also designed and used for a UAV [2]. An adaptive neuro-fuzzy inference system (ANFIS) based controller for a UAV [3] was developed to adjust the altitude, the heading and the speed together. Usually these kind of method are designed for the specific task of control optimization. But there are also some other mathematical approaches that could be adapted to the control task. This is the case of the Cross-Entropy. This is a method that derives its name from the Kullback-Leibler distance, also known as Cross-Entropy distance, which is a fundamental concept of modern information theory. The method was motivated by an adaptive algorithm for estimating probabilities of rare events in complex stochastic networks, which involves variance minimization. The CE method provides a unifying approach to simulate and optimize [4]. Several applications demonstrate the power of the CE method like in [5] for power system reliability evaluation, in [6] for the selection of the right antenna in communication tasks, and in [7] for motion planning. This optimization method was also used for tuning controllers, by the adaptation of the values of a controllers' gains. As far as the authors known the only works presented in the literature using this technique and done by other authors are the optimization of the PID gains of a simulated inverted pendulum by Bodur [8], the optimization of gains of a Fuzzy PD-like controller for cutting force regulation of a drilling process by Haber et al. [9].

The sense-and-avoid problem has been identified as one of the most significant challenges facing the integration of aircraft into the airspace [10]. Here, the term "sense" relates to sensor or sensors used to automatically detect the potential collision, whilst the term "avoid" relates to the autonomous control action used to avoid the potential collision. In the specific case of using a camera to sense the environment this task is called see-and-avoid. The computer vision techniques using either a camera or a thermal image sensor have been used with success in some work presented in the literature as [11][12][13].

In this work is presented the adaptation of the optimization method, not just for the optimization of the gains of the inputs, but also for the optimization of the membership functions' sets of each variable (inputs and outputs), and for the assigned weight for each rule of the base of knowledge of a fuzzy controller. The chosen task for the control approach is the control of the orientation of an unmanned aerial vehicle for see-and-avoid, using the visual feedback information from an onboard camera.

This paper is structured as follows. Section II shows the approach of this method to optimize a fuzzy control approach. The section III shows the vision algorithm used for the detection of the predefined obstacle. The fuzzy control

approach developed for the see-and-avoid task is presented in the section IV. The results of the three optimization processes and the comparison of the optimized and non-optimized controllers in real tests with a quadcopter are presented in section V. Finally, concluding remarks and future work are presented in Section VI.

II. Cross-Entropy Approach for Fuzzy Control Optimization

The Cross-Entropy method is a recent optimization approach developed for stochastic optimization and simulation. It was developed as an efficient method for the estimation of rare-event probabilities. The CE method has been successfully applied to a number of difficult combinatorial optimization problems. This optimization method was used in a set of applications, but most of them dealing with static and noisy combinatorial, and continuous global optimization problems. In this work this method was applied for control optimization, dealing with the specific case of a fuzzy control system. Here, it is shown how to optimize different parameters of a fuzzy controller, such as the gains, the location and size of the membership functions, and also the rules' weight. It is also could be used as a guide for the optimization of other control approaches. A deeper explanation of the Cross Entropy method for general uses is presented on [4].

The algorithm used for the optimization of the different parameters is almost the same with differences in the dimensionality size of the problem to optimize. The optimization of the different parameters could be done in parallel or in series. It was decided to do it in series and following the tuning sequence defined for Fuzzy controllers in [14]. In this work is explained that the changes in the different parts of a FLC structure have different effects to its behavior. These effects goes from macroscopic to microscopic and are easily understood considering the FLC rules' base as a 2D table. Any changes on the gains of the variables affect to the whole table, causing macroscopic effects to the behavior of the controller. This type of tuning or optimization process is also called Scaling Factors (SF) adjustment. If a modification is done only in one set of the membership of one of the variables of the FLC, it will just affects to one row or one line of the table. These effects are measured as medium size effects. Finally if a modification is done to the output or the weight of one rule; it will affects just to one cell of this table, causing microscopic effects to the controller behavior.

A graphical explanation of the optimization process is presented by a flowchart in the Figure 1. This Figure shows the two different loops involve in the optimization process. The central loop corresponds to the phase, where all the controllers generated in each iteration were tested. The external loop is for the updating of the probability density functions, that in this case are the normal distributions.

The generic version of the optimization process for fuzzy controllers is presented in the Algorithm 1. The first step of the algorithm corresponds to the initialization of the CE parameters (the first green box in the Figure 1). The second step corresponds to the generation of all the controllers to

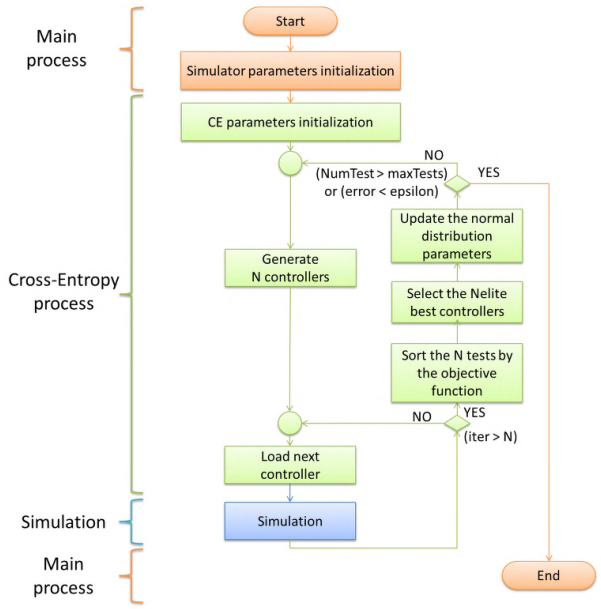


Fig. 1: Flowchart of the Cross-Entropy process for control optimization.

test in the i_{th} iteration, as is shown in the second green box of the principal loop in the Figure 1. In the third step of the Algorithm, all the generated controllers are tested and sort by a cost function. In the fourth step, the probability density function for each component to optimize is updated, based on the best controllers. The steps 2, 3 and 4, are repeat until the ending criterion is reached. Finally, when this ending criterion is reached the optimization process is finished. A more detailed explanation of this Algorithm is presented below.

Algorithm 1 Cross-Entropy Algorithm for Fuzzy controller optimization

1. Initialize $t = 0$ and $v(t) = v(0)$
2 Generate a sample of N controllers: $(x_i(t))_{1 \leq i \leq N}$ from $g(x, v(t))$, being each $x_i = (x_{i1}, x_{i2}, ..., x_i h)$
3. Compute $\phi(x_i(t))$ and order $\phi_1, \phi_2, ..., \phi_N$ from smallest $(j = 1)$ to biggest $(j = N)$.
Get the N^{elite} first controllers $\gamma(t) = \chi_{[N^{elite}]}$.
4. Update $v(t)$ with
$v(t+1) = arg_v min \frac{1}{N^{elite}} \sum_{j=1}^{N^{elite}} I_{\{\chi(x_i(t)) \geq \gamma(t)\}} \cdot ln\, g(x_j(t), v(t))$
5. Repeat from step 2 until convergence or ending criterion.
6. Assume that convergence is reached at $t = t^*$, an optimal value for ϕ can be obtained from $g(., v(t)^*)$.

The adaptation of the general Cross-Entropy method is based on the initial idea that each N samples randomly generated is a single fuzzy controller. The selected parameter to optimize (gains, membership functions or rules' weight) has a set of h components. For example in the case of the gains, h is equal to the number of inputs (gains) to optimize. One probability density function (pdf) is defined for each one of the h components of the selected parameter. The initial

parameters of each pdf depend on the range of the value to optimize. In the case of the membership functions' sets, the value of h depends on the range of the variable itself, and the number of sets that was defined during the designing process of the controller. In case of the rule's weight, a range of $[0,1]$ can be assigned. The probability density function selected in this work was the normal distribution. Based on this preliminary information the Cross-Entropy method was set to generate in each iteration $N = 100$ fuzzy controllers, where each controller x_i is composed by a set of generated values of the h components of the selected parameter to optimize $x_i = (x_{i1}, x_{i2}, ..., x_{ih})$. The $g(x,v) = (g(x_1,v), g(x_2,v), ..., g(x_h,v))$ is the probability function associated to each parameter's component. Depending on the specific parameter to optimize each $x_{i1}, x_{i2}, ..., x_{ih}$ will correspond to the different gains of the variables (inputs and outputs), the position of each membership functions' sets, or to each rules' weight. Once all the generated controllers were simulated, the controllers were sort based on the resulting value obtained by the selected objective function. The most commonly used are, the Integral Time of the Absolute Error (ITAE), the Integral Time of the Square Error (ITSE) or the Root Mean-Square Error (RMSE). Then, only a set of selected best controllers ($N^{elite} = 5$) were used to update the $g(x,v)$. In the presented case, that the selected pdf was a normal distribution, a new mean ($\tilde{\mu}$) and a new sigma ($\tilde{\sigma}$) values were calculated using this set of elite controllers, as is shown in the Equation 1.

$$\tilde{\mu}_{th} = \sum_{j=1}^{N^{elite}} \frac{x_{jh}}{N^{elite}}; \qquad \tilde{\sigma}_{th} = \sum_{j=1}^{N^{elite}} \frac{(x_{jh} - \mu_{jh})^2}{N^{elite}} \tag{1}$$

Where x is the value of the h component of the parameter to optimize that belongs to the j^{th} elite controller in the specific t iteration of the cross-entropy method. The mean vector $\bar{\tilde{\mu}}$ should converge to γ^* and the standard deviation $\bar{\tilde{\sigma}}$ to zero.

In order to obtain a smooth update of the mean and the variance we use a set of parameters (β, α, η), where α is a constant value used for the mean, η is a variable value which is applied to the variance to avert the occurrences of $0s$ and $1s$ in the parameter vectors, and β is a constant value which modify the value of $\eta(t)$, as is shown in the Equation 2.

$$\begin{aligned} \eta(t) &= \beta - \beta \cdot (1 - \tfrac{1}{t})^q \\ \hat{\mu}(t) &= \alpha \cdot \tilde{\mu}(t) + (1-\alpha) \cdot \hat{\mu}(t-1) \\ \hat{\sigma}(t) &= \eta(t) \cdot \tilde{\sigma}(t) + (1-\eta(t)) \cdot \hat{\sigma}(t-1) \end{aligned} \tag{2}$$

Where $\hat{\mu}(t-1)$ and $\hat{\sigma}(t-1)$ are the previous values of $\hat{\mu}(t)$ and $\hat{\sigma}(t)$. The values of the smoothing update parameters are $0.4 \leq \alpha \leq 0.9$, $0.6 \leq \beta \leq 0.9$ and $2 \leq q \leq 7$.

III. Image Processing Front-End

Visual awareness is achieved by using an onboard forward-looking camera. Images from the camera were sent to a laptop Ground Station (GS) for the off-board processing. Then the image is processed and the information extracted is sent to the developed control system approach, that is running in parallel, to finally sent back the specific command to the UAV. The detection and avoidance of the obstacle is done using a visual algorithm based on color detection for the gains optimization process, and based on the detection of an augmented reality (AR) marker for the membership functions' sets, and rules' weight optimizations. The second one is more robust against illumination changes than the first one. For both algorithms the information extracted from the processed images is the location in the horizontal axis of the center of the marker inside the image. This information is used with the camera calibration to estimate the angle between the reference position inside the image, the center of the object and the camera location (the quadrotor). The second algorithm used is based on a developed library of augmented reality using OpenCV named ArUco [15].

IV. Fuzzy Control Approach for See and Avoid

The see-and-avoid strategy is based on the control of the heading or orientation of the aircraft, based on the visual information. The presented control system approach was done by the design and develop of a controller based on Fuzzy Logic. The decision to use this technique is based on the non-linearity of the system, the high sensibility of the UAV against potential disturbances. Furthermore, this technique does not need to use a model of the system to control. The designed heading controller was developed using the Miguel Olivares' Fuzzy Software (MOFS) [16]. The controller was designed as a PID-like, with three inputs and one output. The first input is defined as the angle error between the reference position inside the image, the current position of the object to avoid, and the UAV. The second input is the derivative of this angle estimation, and the third input is the integral on time of this error. The output of the control system approach is the velocity command on degrees per seconds for the orientation change rate of the aircraft. The inputs and the output are defined using triangular membership functions. Different approaches were developed using different number of sets for the input variables. For the gains optimization were used 3 sets and for the optimization of the membership functions and rules' weight were used 5 sets. The numbers of sets of the inputs variables define the size of the rules' base, being in the first case 27 rules and 45 in the second one.

The defuzzification method used in this approach is the height method, shown in Equation 3.

$$y = \frac{\sum_{l=1}^{M} \bar{y}^l \prod_{i=1}^{N} \left(\mu_{x_i^l}(x_i) w_i\right)}{\sum_{l=1}^{M} \prod_{i=1}^{N} \left(\mu_{x_i^l}(x_i) w_i\right)} \tag{3}$$

Where N and M represent the number of inputs variables and total number of rules respectively. $\mu_{x_i^l}$ denote the membership function of the lth rule for the ith input variable. $\bar{y}^l$ represent the output of the lth rule. w_i corresponds to the weight of the ith rule; that could takes values from 0 to 1.

The Figure 2 shows the control loop of the fuzzy control approach and the three optimization processes represented by the numbers $1, 2, 3$ and the related colored lines. The

box defined as objective function corresponds to the specific objective function selected for each optimization (ITSE, ITAE, RMSE, etc).

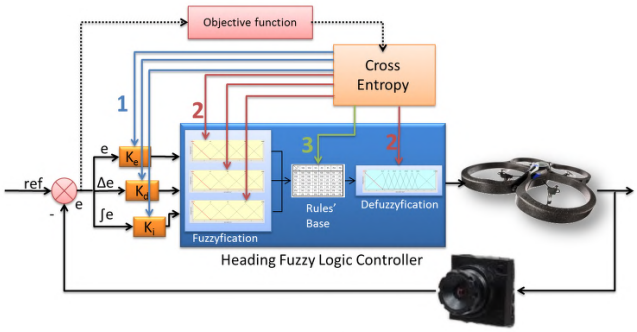


Fig. 2: Vision based control loop for the orientation of the UAV. It is also indicated the three optimization processes done in the different parts of the fuzzy control approach. 1 : Gains optimization, 2 : Membership functions optimization, and 3 : Rules' weight optimization.

V. Experiments

In this section is explained the three different optimization processes and the experiments done for the optimization of the different parts of a fuzzy controller for the specific task of the heading control for avoiding collisions. In the first case is shown the optimization of the gains of the inputs, secondly is shown the optimization process for the membership functions, and finally is shown the optimization of the rules' weight. The optimization of the controller were done in simulated environments. When the optimal controllers were obtained, a set of tests with an AR.Drone quadrotor [17] has been done to corroborate the improved behavior of the optimal controllers. For all the different experiments the testbed was set by, the UAV starts flying against the obstacle and after one meter it starts to do the maneuver to avoid the obstacle. Finally, when is one meter close to the obstacle the maneuver finishes and the UAV has to fly ahead, as is shown in the Figure 3. The longitudinal speed was set in all the simulated tests to 0.3 m/s.

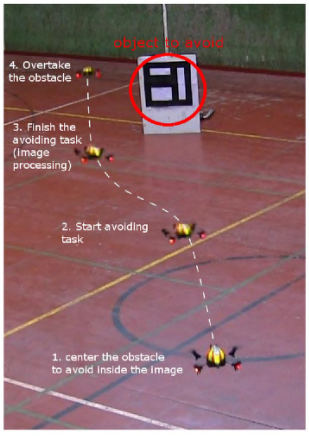


Fig. 3: Description of the avoiding task test.

A. Gains Optimization

This type of optimization is the most simple and the only one that is already presented in the literature done by other authors. In [9] is shown the optimization of the inputs, and output gains of a fuzzy controller for a drilling process in a simulated environment, and under a highly controlled environment such as a drilling machine. In [18] is shown the optimization of the gains of a classic PID controller to command an inverted pendulum in a simulated environment, too. We go a step further doing the optimization of the controller's gains for a such complex and dynamic environment as a UAV control task.

In this case the dimensionality of the problem is very small. It is only three ($h = 3$), one gain for each input. For this experiment we set a simulation environment to do the optimization using the Robotics Operative System (ROS) and the 3D simulator Gazebo [19]. Using this software platform we created a simulated scene composed by a virtual quadrotor based on the Starmac model [20], and a balloon in the middle of the trajectory of the UAV to be the object to avoid. In this case, the objective function used was the ITAE. The initial values for the normal distributions of all the gains were set to $\mu = 0.5$ and $\sigma = 0.5$. After 12 iterations the optimization process converge to all the σ values under 0.016, and setting the gains of the different inputs to 0.9572 for the first input (K), to 0.4832 for the second input (Kd), and to 0.4512 for the third one (Ki). The Figure 4 shows the evolution of the normal distribution associated to one of the gains during the optimization process.

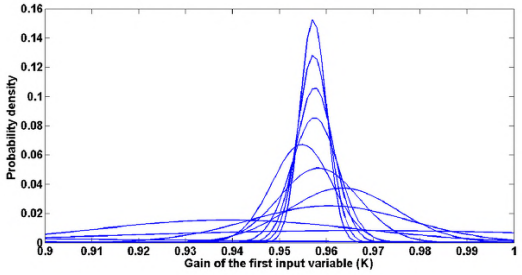
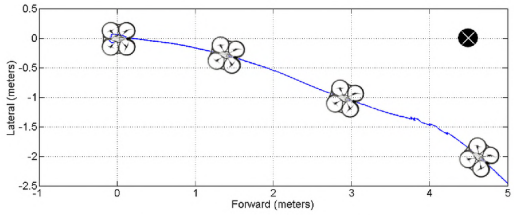


Fig. 4: Evolution of the probability density function for the first input gain.

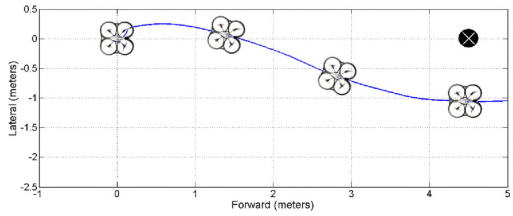
A set of real tests were done at different longitudinal speeds, where the optimized controller shows an improved performance comparing to the non-optimized one (the initial values of μ of the different variables were set at the gains). The Figure 5 shows the performance of the non-optimized controller and the optimized controller, and how the non-optimized controller can not accomplish the task successfully, and the optimized controller can do it in the right way.

The different tests done to check the behavior of the optimized controller versus the non-optimized are shown as a boxes graphs in the Figure 6. This Figures shows the results measured by the RMSE in degrees of the full length of the test at different longitudinal speeds.

A detailed information of this specific optimization pro-

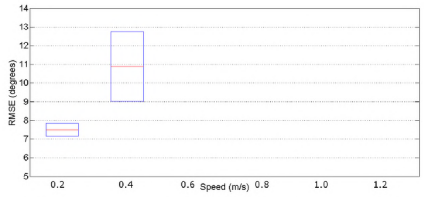


(a) Non-optimized controller

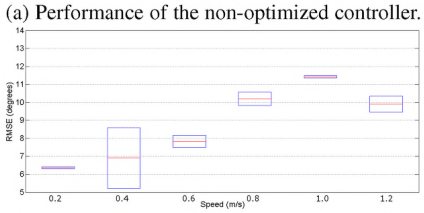


(b) Optimized controller

Fig. 5: 2D reconstruction of the trajectory with a longitudinal speed equal to 0.08 m/s, where the black circle with the white x represents the location of the object to avoid.



(a) Performance of the non-optimized controller.



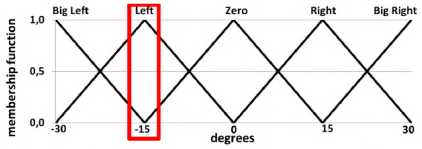
(b) Performance of the gains optimized controller.

Fig. 6: Performance of the gains optimized controller versus the non-optimized one for the avoiding collision task with a real quadrotor at different longitudinal speeds.

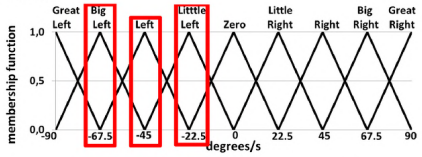
cess, and all the tests done are shown in [21]. It must be taken into account that a mistake occurs in the published version about the speed measurements by the inclusion of an extra zero, that means, a longitudinal speed of 0.03 m/s really represents 0.3 m/s.

B. Membership Functions Optimization

The optimization of the membership functions' sets of all the inputs and the output consists in the modification of the value of center of the each set. It corresponds to the left range of the next set, and the right range of the previous set, as is shown in the Figure 7. In this Figure the first inputs and the output are represented. The second and the third inputs are very similar to the first just changing the range of the variable from $-30, 30$ to $-10, 10$. The symmetric definition of the variables, and the fixed values of the full variable range and the center, reduces the points (components) to optimize to one for each input and three for the output. For this optimization process the dimension of the problem is six ($h = 6$), that means, one set for each input (the second set), and three sets for the output (the third, fourth and fifth sets). Comparing to the previous optimization case the number of sets for each variable was modified from three to five, to check if the previous definition was the correct one of not for this specific problem.



(a) **First input** membership functions: Angle error.



(b) **Output** membership functions, Heading Command.

Fig. 7: Definition of the Fuzzy PID-like controller to Optimize. The values to optimize are marked by a red rectangle.

For this optimization case and the next one a different simulation environment was defined using Matlab. With this software there is no 3D representation for each test, so it reduces drastically the CPU power consumption and time elapsed for each test. Must be taken in to account that in both simulation environments the model used is just an rough approximation of the real one.

The initial values of the μ and the σ of each pdf depend on the range of the variables, being the $\mu_0 = \frac{set's\ size}{2}$, and $\sigma_0 = (set's\ size)\frac{2}{3}$, e.g in the input shown in the Figure 7a, the initial value of the μ is -15 and the σ is 10. The Figure 8 shows the evolution of one of the six sets to optimize.

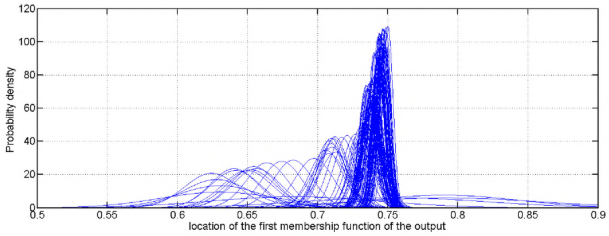


Fig. 8: Evolution of the probability density function for one of the set location and size.

This optimization process reveals that five sets were too much for some of the variables, more exactly for the derivate and the integral of the error. For these variables the new position of the second set overlaps the first set, so it means that one set must be canceled and reduced from five to three. This sets cancellation reduces the size of the rules base, from 125 to 45 rules. For the first input, the center of the second set

was modified from -15 (as is shown in Figure 7a) to -3.18, being the same for the right side of the variable definition. For the second and the third inputs, the center of the second set was modified from -5 to -10, this value reveals the overlapping sets. The center of the three sets of the output to optimize goes from -67.5, -45, and -22.5 (as is shown in the Figure 7b) to -74.74, -56.4, and -15.54 respectively. The new configuration of the variables modify the behavior of the control system improving the response for the avoiding tasks. This optimization process takes 90 iterations to get the final values for the sets.

After this optimization process, the optimization of the rules' weight was done. This optimization process and some results of the reals test of both optimization process are shown in the next subsection.

C. Rules' Weight Optimization

The last application of the CE optimization method presented in this work affects to the rules' weight. Commonly the base of rules of a fuzzy logic controller has no weights assigned, but it could be used to assign more importance to some rules against others. In this case we associate a weight value for each rules. After the previous optimization phase, the rules base is composed by 45 rules. Because of the symmetric definition of the control system, each rule has an opposite rule associated. This symmetry ensures the the controller will has the same behavior to solve the avoiding task either is the object is at the left side or at the right side. Because of it we only have to optimize the number of the rules plus one divided by two. It means that the dimensionality of this optimization problem is reduced to $h = 23$. So there are 23 components to optimize with 23 pdf associated in this process. We have assigned an initial value equal to 0.5 to each rule, that is the μ of the normal distribution of each rule. There is also a definition of the maximum weight value that was set equal to one. The Figure 9 shows the evolution of one of the rules' weight.

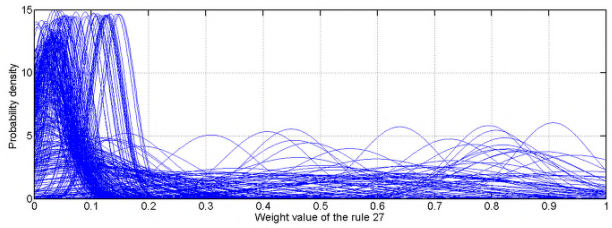


Fig. 9: Evolution of the probability density function for one of the rules' weight.

This optimization process reveals that some of the implemented rules were not necessary for this specific task, by the reduction of the weight to a value under 0.01. By the cancellation of these rules the rules base was reduced from the 45 rules of the previous optimization phase until 31 rules.

A comparison between the non-optimized controller, and the two optimized controller for the membership functions' and the rules' weight is shown in the Figure 10. This Figures represents the evolution of the control in a real test. The reference was changed in one precise moment from zero to 20 degrees, but it is not a real step signal because the UAV continues flying with a constant speed against the object, so the reference is changing in each new frame acquisition. In order to be more easy to understand we represent it as a step signal.

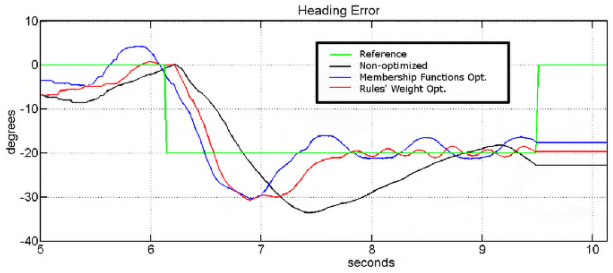
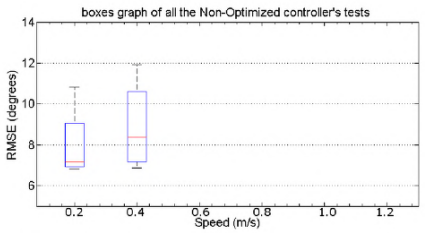
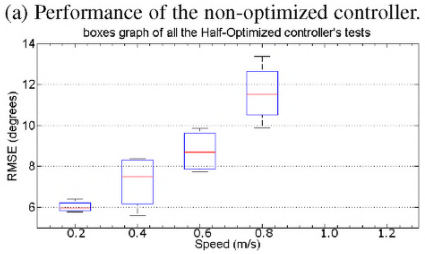


Fig. 10: Comparison of the non-optimized controller and the membership function and rules optimized controllers by the evolution of the error in a real test.

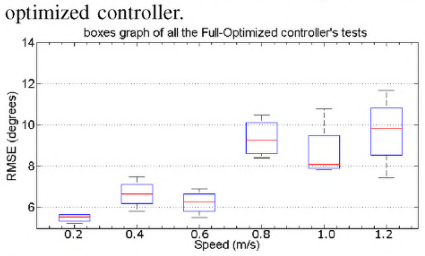
A large set of tests have been done to check the behavior of the two optimized controllers versus the non-optimized one. The Figures 11 shows the results represented by boxes graphs and measures by the RMSE in degrees of the full length of the test at different longitudinal speeds from 0.2 to 1.2 meters per second.



(a) Performance of the non-optimized controller.



(b) Performance of the membership functions optimized controller.



(c) Performance of the rules' weight optimized controller.

Fig. 11: Performance of the three controller for the avoiding collision task with a real quadrotor at different longitudinal speeds.

It must be taken into account that the quadrotor used for all the presented tests has not a onboard computer, so the image processing must be done in a remote computer connected by WiFi with the UAV. The connection is quite unstable an the number of images per seconds received is not constant during the tests. This value moves from 6 to 20 frames per seconds. Some video of the presented tests are available on [22], [23], [24],.

VI. Conclusions and Future Work

In this paper, we have presented a novel approach of the Cross-Entropy method to optimize three different parts of a fuzzy controller making easier the tuning of a basic fuzzy controller for a specific task. A fuzzy control approach to command the orientation of an unmanned aerial vehicle for avoiding collisions was used for the optimization process. The optimization of the inputs' gains, the membership functions' sets, and the rules' weight were done using this recently developed optimization method. The processes of the different optimizations were done in two different simulated environments. The optimized controllers were compared with the initial controller configurations at different speeds in indoor tests with a real quadrotor. The results show that the optimized controllers improve the behavior of the non-optimized ones, by a big reduction of the root mean square error of the full tests, and by finishing successfully the avoiding task at higher speeds. The membership functions and the rules' weight optimization processes, also reduce the number of the rules in the base of knowledge, simplifying the initial control system approach.

The presented adaptation of the Cross-Entropy method for fuzzy control optimization could be also used to optimize other control approaches, that is in what the authors are working right now, to compare the different optimized control approaches for different specific tasks.